Achieving Flexibility in LDPC Code Design by Absorbing Set Elimination

Jiajun Zhang*, Jiadong Wang*, Shayan Garani Srinivasa†, Lara Dolecek*
*Department of Electrical Engineering, University of California, Los Angeles
†Western Digital Corporation
Email: zjj1990228@ucla.edu, wjd@ee.ucla.edu, shayan.garani@wdc.com, dolecek@ee.ucla.edu

***Abstract*—Low-density parity-check (LDPC) codes are attractive since their performance is known to approach the Shannon limits for suitably large block lengths. However, for moderate block lengths, error floors still jeopardize the performance even of well-designed LDPC codes. Previous work has shown that the error floor of a broad class of LDPC codes is due to certain graphical structures called absorbing sets. Separable, circulant-based (SCB) codes represent a general family of high-performance, hardware-friendly LDPC codes built out of circulants. A recently proposed technique applies row selection and column elimination methods to SCB codes to dramatically decrease error floors by avoiding certain small dominant absorbing sets in a principled way. This paper focuses on improving the greedy column elimination method to achieve greater flexibility in code rate while provably avoiding small dominant absorbing sets. Flexibility and low implementation complexity are therefore possible without sacrificing SCB code performance.**

I. Introduction

Low-density parity-check (LDPC) codes are well-known for their capacity-approaching property with iterative decoding [1]. However, for moderate block lengths and low frame error rates (FERs), LDPC codes often have an error floor, reflected in the flattening of the FER performance curves. This error floor is attributed to the sub-optimality of message passing decoding algorithm in Tanner graphs with cycles. For additive noise channels and practical decoder implementations, prior research has linked the primary reason for this sub-optimality and the resultant error floor with certain substructures of the Tanner graph called absorbing sets [2]. An absorbing set is a particular type of a near-codeword or a trapping set guaranteed to be stable under bit-flipping operations. Several techniques have been recently proposed to improve the absorbing set (trapping set) spectrum. These techniques include adding redundant rows to the parity check matrix, elimination of small cycles, increasing the girth, and a careful re-wiring of edges in the Tanner graph, e.g., [3], [4], [5], [6], and [7].

Separable, circulant-based (SCB) codes constitute a general family of high-performance, hardware-friendly LDPC codes built out of circulants. These codes can each be constructed by shortening the so-called mother SCB code whose parity check matrix rows are each comprised of a concatenation of all possible circulants.

A recently developed approach [7] offers a deterministic procedure for row and column selection that provably eliminates small dominant absorbing sets for SCB codes. A careful re-ordering of the constituent circulants in the parity check matrix amounts to a "re-wiring" of the edges in the Tanner graph. This approach therefore preserves code structure, node degree and the ease of hardware implementation while offering provably improved performance.

In [7], the row selection is performed based on congruential equations describing dominant absorbing sets, and is further simplified by additive and multiplicative invariance of circulant shifts. Depending on the bit node degree, a proper row selection is in itself sufficient to eliminate dominant absorbing sets [7], [8]. In certain cases, however, an additional column selection must be performed to eliminate remaining absorbing sets. The approach in [7] for the column selection is a greedy algorithm that selects a subset of columns from the set of all possible columns of the parity check matrix of the mother SCB code. An excessive elimination of a large subset of candidate columns implies an overly conservative upper bound on the achievable code rate. Simply increasing the inner circulant matrix dimension can increase the code rate, but it also increases the block length, and in turn the decoding complexity.

The contributions of this paper are as follows. We relate the column elimination problem to a well-defined, NP-complete problem in theoretical computer science called *the hitting set problem* [9], and we develop new algorithms that improve upon the greedy column elimination technique offered in [7]. Our approach to eliminate fewer columns while provably avoiding small dominant absorbing sets thus offers an enlarged design space of code parameters while preserving target performance and complexity.

Since the hitting set problem is NP-complete, we conclude that in general, there is no efficient method to find an optimal solution to the column selection problem. For moderately small block lengths, we present an efficient way to obtain the optimal solution to column selection using binary linear programming (BLP). For large block lengths we provide a polynomial-time search algorithm whose solution sets improve that of the greedy algorithm.

Section II summarizes the main mathematical objects, including (mother) SCB codes, absorbing sets as well as previous results on smallest absorbing sets in column weight-4 SCB codes. Section III discusses the NP-complete nature of the column selection problem, and proposes new, simple algorithms with results provably better than the greedy approach used in [7] for column weight-4 SCB codes. Section IV provides simulation results that demonstrate the improvement in the code design flexibility for comparable code performance. Section V delivers the conclusions.

II. Definition and Preliminaries

We now quickly summarize SCB codes, absorbing sets and the relevant previous results regarding the elimination of small absorbing sets in SCB codes.

A. *SCB codes and absorbing sets*

Circulant-based LDPC codes are a family of structured regular LDPC codes with r variable nodes and c check nodes, and are particularly amenable for high-throughput implementation [10].

The parity check matrix of a circulant-based LDPC code is:

$$H_{p,f}^{r,c} = \begin{bmatrix} \sigma^{f(0,0)} & \sigma^{f(0,1)} & \sigma^{f(0,2)} & \dots & \sigma^{f(0,c-1)} \\ \sigma^{f(1,0)} & \sigma^{f(1,1)} & \sigma^{f(1,2)} & \dots & \sigma^{f(1,c-1)} \\ \sigma^{f(2,0)} & \sigma^{f(2,1)} & \sigma^{f(2,2)} & \dots & \sigma^{f(2,c-1)} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \sigma^{f(r-1,0)} & \sigma^{f(r-1,1)} & \sigma^{f(r-1,2)} & \dots & \sigma^{f(r-1,c-1)} \end{bmatrix}, \tag{1}$$

where σ is a $p \times p$ circulant matrix.

A column (row) of inner circulants is called column (row) group. Each variable node has a label (j,k) where $j \in \{0,...,c-1\}$ is the index of the corresponding column group and $k \in \{0,...,p-1\}$ is the actual index of the column in that column group. Similarly, each check node has a label (i,l) with $i \in \{0,...,r-1\}$ and $l \in \{0,...,p-1\}$.

SCB codes are defined as follows.

Definition 1 (Separable, Circulant-Based (SCB) Code): An SCB code is a circulant-based LDPC code with a parity-check matrix $H_{p,f}^{r,c}$ in which $f(i,j)$ is separable, i.e., $f(i,j) = a(i) \cdot b(j)$. ■

Specific parity-check matrices of different SCB codes can be viewed as originating from the same mother SCB matrix $H_{p,f_m}^{p,p}$ with $f_m(i,j) = i \cdot j$. The functions $a(i)$ and $b(j)$ each with range $\{0,...,p-1\}$ specify which rows and columns are selected from the SCB mother matrix. The SCB code whose parity check matrix is the mother SCB matrix is called mother SCB code. Many well-known quasi-cyclic LDPC codes, including [11] [12], are special cases of the SCB codes.

In addition to providing the parity check matrix H, an LDPC code is often described in terms of its Tanner (bipartite) graph $G_H = (V,F,E)$ where V denotes the set of the variable nodes, F denotes the set of the check nodes, and E is the set of edges between variable and check nodes.

In the Tanner graph G_H, an absorbing set is defined as follows [2].

Definition 2: Consider a subset V_a of V. Let $G_a = (V_a, F_a, E_a)$ be the bipartite graph compromised of the set V_a, the neighboring check nodes described by the set F_a, and the edge set E_a connecting V_a and F_a. Let $o(V_a) \subset F_a$ denote the neighbors of V_a that have odd degree with respect to G_a and let $e(V_a) \subset F_a$ denote the neighbors of V_a that have even degree with respect to G_a. If $|o(V_a)| < |e(V_a)|$, the subgraph $G_a = (V_a, F_a, E_a)$ of G_H is an (a,b) absorbing set.

B. $(6,4)$ *absorbing sets of* $r = 4$ *SCB codes*

A result in [2] proves that $(6,4)$ absorbing sets are the smallest possible absorbing sets for SCB codes with $r = 4$ for $p > 19$, and that these $(6,4)$ absorbing sets can be classified into three non-isomorphic configurations. In [7], it was proven that, after a careful selection of the four row groups from the mother SCB matrix, two out of three non-isomorphic $(6,4)$ structures can be provably eliminated, and it was also proven that the remaining structure cannot be eliminated by a row selection alone. This remaining configuration is depicted in Figure 1. Here, black circles represent the set V_a, white squares represent the set $e(V_a)$, and black squares represent the set $o(V_a)$. Note that the size of V_a is 6 and that the size of $o(V_a)$ is 4.

After an appropriate row selection $a(i)$ was applied to produce so-called selected-row (SR) SCB codes and their matrices $H_{p,a(i)\cdot j}^{4,p}$, the removal of a few column groups was then considered in order to ensure that all $(6,4)$ absorbing sets are avoided in a way that does not introduce any new smaller absorbing sets. This shortened SR SCB matrix is denoted as $H_{p,a(i)\cdot b(j)}^{4,p}$, where $b(j)$ is the column selection function.

III. Improved column selection algorithm for flexible code design

In this section we describe new (approximate) branch-and-bound (BNB) algorithms that efficiently identify the column selection function $b(j)$, applied to a mother SCB matrix $H_{p,f_m}^{p,p}$, a matrix that has p column groups. We show that the column selection problem is NP-complete. We summarize the greedy column elimination algorithm proposed in [7], and use it as an upper bound for the proposed search algorithm.

It is convenient to map each of the $(6,4)$ absorbing sets (shown in Figure 1) into a 6-element set, denoted by $S_i = \{j_1, j_2 ..., j_6\}$, where j_l is the column group index of the l-th variable node, denoted as v_l in Figure 1, in the corresponding i-th absorbing set. Note that the total number of these sets is $\Theta(p^3)$, see [2] for the exact cardinality discussion.

Suppose V is the set of all column group indices of the eliminated variable nodes, where $V \subset T = \{0,1,...,p-1\}$. A careful selection of eliminated columns provably avoids all $(6,4)$ absorbing sets of this configuration by ensuring that $V \cap S_i \neq \emptyset$ for every i. The final parity check matrix then has columns from $T \setminus V$ (which effectively specifies the mapping $b(j)$ for j denoting the column index.) The greedy algorithm [7] updates the set V on a per-element basis to ensure non-zero intersections so that at any given time the largest number of absorbing sets are eliminated.

First, we note that finding the smallest number of columns to cut in order to avoid all small dominant absorbing sets can be viewed as a particular case of a famous NP-complete problem, known as the hitting set problem [9]. The hitting set problem is concerned with finding a minimal set that has a non-zero intersection with a collection of sets. Since

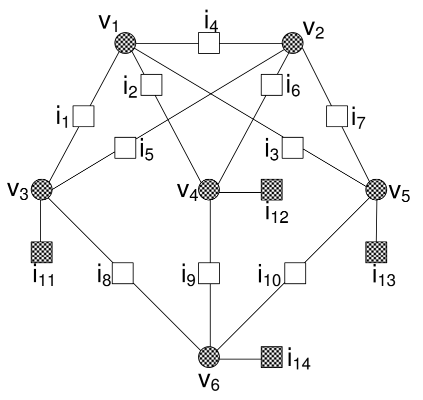

Fig. 1: Depiction of a $(6,4)$ absorbing set configuration.

the column selection problem falls in the NP-complete space, there is no efficient, polynomial-time, algorithms to solve it. We now propose branch-and-bound (BNB) algorithms. When the dimension p of the inner circulants is small the proposed algorithm finds the optimal solution, otherwise an approximate BNB approach is used.

For moderately small block lengths, the column selection problem is translated into a binary linear programming (BLP) problem. Define an indicator binary matrix $M_{|S|\times p}$ where, for a given Tanner graph G_H, S is the collection of all distinct absorbing sets S_i of type shown in Figure 1 (for $1 \leq i \leq |S|$), and p is the dimension of the inner circulants, as well as being the total number of column groups in a mother SCB matrix. The entry $M_{i,j}$ is equal to 1 if the j-th column group belongs to the absorbing set S_i. Consider a binary column vector $X = (x_1, x_2, ..., x_p)^T$, where $x_j = 1$ if the jth column group needs to be eliminated from the SR SCB matrix to avoid small dominant absorbing sets.

In order to produce the set of eliminated columns with the smallest size, we need to minimize the number of ones in X, i.e., $||X||_1$. With the constraint that the minimum entry of MX is bigger than 1, in each absorbing set S_i there exists at least one variable node whose column group belongs to the cut column group list. This constraint limits X to be a valid column selection that eliminates all small dominant absorbing sets. This BLP problem is summarized as follows:

$$\begin{array}{rl} \min & ||X||_1 \\ \text{s.t.} & MX \geq B \\ & x_i = 0 \text{ or } 1 \text{ for } i \in \{1,2,...,p\} \end{array} \tag{2}$$

where $B^T = (1,...,1)$ is a vector of length $|S|$ with all elements equal to one.

For very large block lengths, due to the NP-complete nature of this problem, it is impossible to find an optimal solution in polynomial time. A previous result in [13] shows an efficient branch-and-bound (BNB) algorithm for the BLP problem. Combining this BNB approach with the concept of a generic approximate algorithm [9], we develop an approximate BNB algorithm to the column selection problem.

We now identify a lower and an upper bound to a solution offered by the approximate BNB algorithm to the BLP problem above. Since B is a vector of length $|S|$ with all ones, we can simply write $||B||_1 = |S|$. With all entries in MX and B being positive, taking 1-norm of both sides in the expression (2) preserves the inequality. Thus the solution to the BLP problem described by equation (2) has a lower bound $|S|/\,||M||_1$. The upper bound of the column selection problem is calculated by the greedy algorithm previously proposed in [7], denoted as function *UpperBound*() in Algorithm 1. Note that this greedy algorithm has linear time complexity.

The approximate BNB algorithm sequentially simplifies the column elimination problem such that at each state of the problem, the label (A, T_{in}, T_{out}) indicates a sub-problem of the original BLP problem. Here A is the sub-matrix of the matrix M in equation (2), while T_{in} and T_{out} are the sets of columns selected to be cut and not to be cut, respectively. Note that T_{in} and T_{out} are disjoint subsets of the column set $T = \{0, 1, ..., p-1\}$. The initial state is $(M, \emptyset, \emptyset)$. A state is a final state if the associated matrix A has exactly one column (row) with non-zero entries. The function that tests whether the current state is a final state is called *FinalState*() in Algorithm 1.

For each state (A, T_{in}, T_{out}), we update our solution if the upper bound of the state solution, calculated by *UpperBound*(), is a better solution. If the state is not a final state, we find a column c in A with the maximum 1-norm. We then split (A, T_{in}, T_{out}) into $(A_1, T_{in} \cap \{c\}, T_{out})$ and $(A_1, T_{in}, T_{out} \cap \{c\})$ where A_1 is the sub-matrix of A obtained by cutting the splitting column c from A. This splitting method is denoted as function *Split*() in Algorithm 1. The splitting process is terminated if either all states have been traversed or the lower bound $|S|/\,||M||_1$ is reached. The above process produces a binary tree with each node of the tree representing a state of the BNB algorithm.

The number of layers in the tree is up to p, corresponding to a time complexity of $\Theta(2^p)$ in the BNB algorithm. In order to make this a polynomial-time algorithm, we utilize the generic concept introduced in [9]. For each column of A in the original BLP problem, we keep a reputation value, initialized as all 0's. If cutting that column in a given state results in a better solution in a descendant state, we increase the column's reputation by one. This reputation updating mechanism is called *UpdateReputationList*() in Algorithm 1. We then introduce an approximating parameter f. For a binary tree with up to p layers, we use the regular BNB algorithm to build up our reputation list in the first f layers. Starting from the $(f+1)$st layer, if the reputation of a splitting column c is strictly positive, we reduce the state (A, T_{in}, T_{out}) into $(A_1, T_{in} \cap \{c_j\}, T_{out})$, else into $(A_1, T_{in}, T_{out} \cap \{c_j\})$. This updated version of the function *Split*() is called *ReputationReduction*() in Algorithm 1. Since the reputation updates are completed in at most p layers, the approximate BNB algorithm runs in polynomial time. Note that for moderate p (e.g., $p \leq 37$ in our simulations), we can set $f = p$ to run the exact BNB algorithm without any approximation to get the optimal solution.

The detailed algorithm is shown in Algorithm 1.

Algorithm 1 Approximating branch-and-bound algorithm for the column selection problem

```
$P_{original} \leftarrow (M\emptyset, \emptyset)$
$Tree \leftarrow \{P_{original}\}$
$G_{sol} \leftarrow UpperBound(P_{original})$
$F_{tree} \leftarrow \{(P_{original}, G_{sol})\}$
**while** $Tree \neq \emptyset$ **do**
    choose $\gamma = (A, T_{in}, T_{out}) \in Tree$;
    $Tree \leftarrow Tree/\{\gamma\}$;
    **if** $|UpperBound(\gamma)| < |G_{sol}|$ **then**
        $G_{sol} \leftarrow UpperBound(\gamma)$;
    **end if**
    $UpdateReputationList(\gamma, F_{tree})$;
    **if** $FinalState(\gamma) = FALSE$ **then**
        **if** $|T_{in} \cap T_{out}| < f$ **then**
            $(\gamma_1, \gamma_2) = Split(\gamma)$;
            $Tree \leftarrow Tree \cap \{\gamma_1, \gamma_2\}$;
        **else**
            $\gamma_1 = ReputationReduction(\gamma)$;
            $Tree \leftarrow Tree \cap \{\gamma_1\}$;
        **end if**
    **end if**
**end while**
```

IV. Results

In this section we experimentally demonstrate an improvement in the code rate flexibility obtained with the proposed search algorithms. In the simulations, we use 200 iterations and $Q4.2$ fixed-point quantization with 4 bits for integer part and 2 bits for fractional values. The implemented decoding algorithms is SOFT-XOR [14].

Achievable design choices obtained from the exact BNB algorithm and the approximate BNB algorithm are collected in Figure 2 along with choices generated by the greedy algorithm in [7]. Note that the greedy algorithm offers the optimal solutions for small p's, such as $p = 17$. As the parameter p increases, the solution set generated by the greedy algorithm is no longer optimal. For a fixed block length, and as shown by the vertical line in Figure 2, the new algorithms offer shortened SR SCB codes with higher code rates. At the same time, for a target code rate, and as shown by the horizontal line in Figure 2, the new algorithms offer shortened SR SCB codes with smaller block lengths. For example, for the block length of around 1700 bits, the original greedy algorithm can only produce an shortened SR SCB code with the code rate 0.8261, while the new BNB algorithm offers codes with code rate choices ranging from 0.8261 to 0.8462. For a targeted code rate 0.8950, the new achievable code designs have block lengths ranging from 3783 bits to 4953 bits, while the greedy method requires the block length (of at least) 4953 bits.

In Table I, we compare the performance of SR SCB codes with greedily eliminated columns with that of SR SCB codes with optimally eliminated columns for check node degree $p = 23$ and bit node degree $r = 4$. We can observe that the

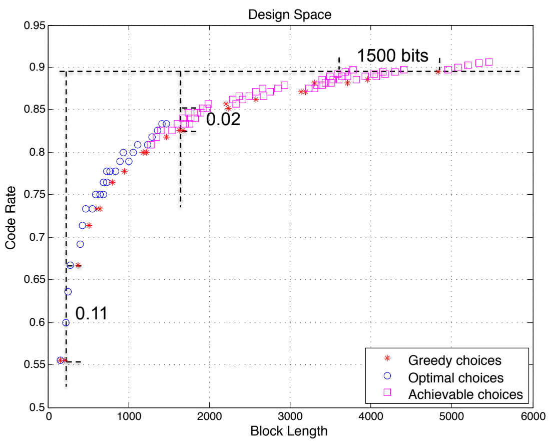

Fig. 2: Design space for shortened SR SCB codes with $r = 4$.

SNR	FER (Greedy)	FER (Optimal)
4.0	5.383×10^{-4}	4.993×10^{-4}
4.4	8.699×10^{-5}	9.771×10^{-5}
4.8	1.824×10^{-5}	2.056×10^{-5}
5.2	4.965×10^{-6}	4.373×10^{-6}
5.6	1.163×10^{-6}	1.265×10^{-6}

TABLE I: Performance comparison of proposed shortened SR SCB codes, $r = 4$ and $p = 23$ with AWGN fixed point decoder under optimal elimination, $N = 299$, rate=0.6667, and greedy elimination, $N = 209$, rate=0.5556.

performances are approximately the same. Yet, the new design offers a range of possible code lengths up to 50% higher, and up to 20% higher rate than the single-point solution offered by the greedy approach. Note that if one wishes to design a code with code length of around 300 bits and the rate of around 0.67, the greedy approach would require an increase in the size p of the constituent circulants, thus effectively increasing the implementation complexity relative to the proposed method.

In Figure 3, we compare the performance of a shortened SR SCB code with greedily eliminated columns with that of a shortened SR code with columns eliminated using the approximate BNB algorithm, for bit node degree $r = 4$ and block length of around 2300 bits. These two codes have the same performances even though the code generated by the approximate BNB algorithm has higher code rate. The decoder complexities of the two codes are comparable since the check node degrees are similar.

We also compare the performance of a shortened SCB code generated by the approximate BNB algorithm with randomly-constructed circulant-based LDPC codes with same code rate and block length. These random codes are constructed such that their girths are guaranteed to be larger than 4 and such that their parity check matrices are two-dimensional arrays of constituent circulants (so that the implementation complexity would be comparable). The performance of the new code is at least an order of magnitude better than the randomly constructed circulant-based codes with girth larger than 4.

Table II shows the error profiles of these three types of codes at two representative SNR points. Note that $(4, 4)$ absorbing sets are dominant absorbing sets in the error floor of random

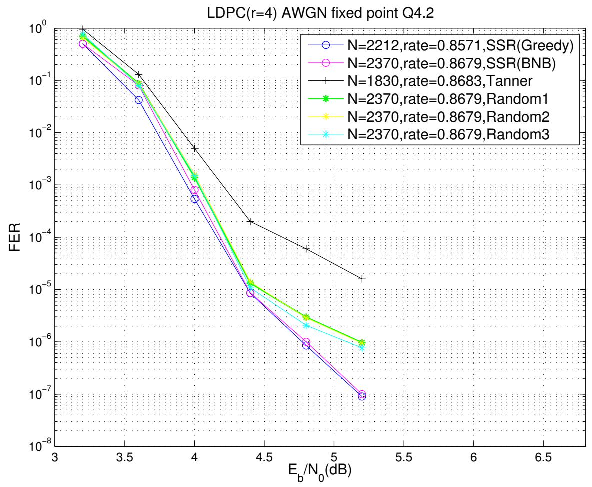

Fig. 3: Performance comparison of shortened SR LDPC codes generated by the greedy algorithm and the new branch-and-bound approximating algorithm, and three random circulant-based LPDC codes with girth larger than 4, and an LDPC code from [15].

Code	SNR	n.e	$(4,4)$	$(5,4)$	$(6,2)$	$(6,4)$	$(8,2)$
Tanner	4.8dB	300	247	1	3	43	1
RC1	4.8dB	300	190	7	38	25	1
RC2	4.8dB	170	100	9	0	20	1
RC3	4.8dB	120	68	13	0	6	1
SSR	4.8dB	130	0	0	0	0	67
SSR(BNB)	4.8dB	143	0	0	0	0	73

Code	SNR	n.e	$(4,4)$	$(5,4)$	$(6,2)$	$(6,4)$	$(8,2)$
Tanner	5.2dB	40	20	0	10	1	0
RC1	5.2dB	35	26	1	1	3	0
RC2	5.2dB	35	24	3	1	4	1
RC3	5.2dB	35	23	1	1	5	0
SSR	5.2dB	30	0	0	0	0	27
SSR(BNB)	5.2dB	35	0	0	0	0	33

TABLE II: error profiles for an shortened SR (SSR) SCB code generated by approximate BNB algorithm, 3 random circulant codes with girth larger than 4, and a code from [15]. The number of errors collected is n.e.

circulant codes and a code from [15].

The performance improvement is thus achieved by a structural elimination of all dominant absorbing sets smaller than or equal to $(6,4)$ absorbing sets.

In summary, note that if the absorbing sets are provably eliminated from the Tanner code of the graph, they are also eliminated from *all* Tanner graphs corresponding to the punctured transformation of the code. The results above therefore show the the improvement in the solution to the column selection problem will lead to a greater flexibility in the code design and target code rate without sacrificing performance and decoding complexity. The approximate BNB algorithm also delivers codes with performances of at least 1 magnitude better than random circulant codes with girth larger than 4. Based on experiments in [8], the result can be easily extended to other check node degrees.

V. Conclusion and Future Work

This paper introduces a novel approach to calculate the minimum set of columns needed to be cut to eliminate all small dominant absorbing sets in SCB codes. The proposed BNB algorithm solves the column selection problem exactly for moderate block lengths. For large block lengths, a polynomial-time BNB algorithm delivers suboptimal solutions better than the previously proposed greedy algorithm [8]. The new algorithms can be easily implemented for larger block lengths and higher check node degrees. The new algorithms provide a way of achieving flexibility in code rate and block length for shortened SR SCB codes without compromising code performance and decoder architecture complexity. Future work includes parallelizing the BNB algorithm over a polynomial number of processors for added improvements.

Acknowledgement

This work was supported in part by a research grant from UC Discovery and Western Digital.

References

[1] T. Richardson, M. Shokrollahi, and R. Urbanke, “Design of capacity-approaching irregular low-density parity-check codes,” *IEEE Trans. on Info. Theory*, 47(2):616-637, Feb. 2001.
[2] L. Dolecek, Z. Zhang, M. J. Wainwright, V. Anantharam, and B. Nikolic, “Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes,” *IEEE Trans. on Info. Theory*, 56(1):181-201, Jan. 2010.
[3] S. Laendner, T. Hehn, O. Milenkovic, and J. Huber, “When does one redundant parity-check equation matter?” In *Proc. IEEE GLOBECOM*, San Francisco, CA, Nov. 2006.
[4] O. Milenkovic, N. Kashyap, and D. Leyba, “Shortened array codes of large girth,” *IEEE Trans. on Info. Theory*, 5(8):3707-3722, Aug. 2006.
[5] R. Asvadi, A. H. Banihashemi, and M. Ahmadian-Attari, “Lowering the error floor of LDPC codes using cyclic liftings,” *IEEE Trans. on Info. Theory*, 57(4):2213-2224, Apr. 2011.
[6] D. V. Nguyen, B. Vasic, and M. Marcellin, “Structured LDPC codes from permutation matrices free of small trapping sets,” In *Proc. of IEEE Info. Theory Workshop (ITW)*, Dublin, Ireland, Sep. 2010.
[7] J. Wang, L. Dolecek, R. Wesel, “Controlling LDPC absorbing sets via the null space of the cycle consistency matrix,” In *Proc. IEEE Int. Conf. on Comm. (ICC)*, Kyoto, Japan, June 2011.
[8] J. Wang, L. Dolecek, Z. Zhang, and R. Wesel, “Absorbing Set Spectrum Approach for Practical Code Design,” In *Proc. Int. Symp. on Info. Theory (ISIT)*, Saint-Petersburg, Russia, August 2011.
[9] H. Brönnimann and M. T. Goodrich, “Almost optimal set covers in finite VC-dimension,” *Discrete and Comput. Geometry*, 14(4):463-479, 1995.
[10] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, and M. J. Wainwright, “Design of LDPC decoders for improved low error rate performance: quantization and algorithm choices,” *IEEE Trans. on Comm.*, 57(11):3258-3268, Nov. 2009.
[11] J. L. Fan, “Array-codes as low-density parity-check codes,” In *Proc. of Second Int. Symp. on Turbo Codes and Iter. Info. Proc. (ISTC)*, Brest, France, Sep. 2000.
[12] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, “LDPC block and convolutional codes based on circulant matrices,” *IEEE Trans. on Info. Theory*, 50(15):2966-2984, Dec. 2004.
[13] A. Fijany and F. Vatan, “New high-performance algorithmic solution for diagnosis problem,” In *IEEE Aerospace Conf. (IEEAC)*, 3863-3873, Mar. 2005.
[14] M.M. Mansour and N.R. Shanbhag, “High-throughput LDPC decoders,” *IEEE Trans. on VLSI Systems*, 1(6):976-996, Dec. 2003.
[15] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, “LDPC block and convolutional codes based on circulant matrices,” *IEEE Trans. on Info. Theory*, 50(15):2966-2984, Dec. 2004.